NOVIA
UNIVERSITY OF APPLIED SCIENCES

Product data transfer between global and local systems

Niklas Järnström

Bachelor's thesis in Information Technology

Vasa 2013

BACHELOR'S THESIS

Author:	Niklas Järnström
Degree Programme:	Information Technology, Vasa
Supervisor:	Ray Pörn

Title: *Product data transfer between global and local systems*

Date: 21.03.2013 Number of pages: 34 Appendices: 3

Abstract

This thesis is commissioned by Wärtsilä Finland Oy Ship Power and is about transferring product data from the global application Teamcenter into a local database called TERPS. The purpose is to expand the PDM (Product Data Management) to cover also project deliveries or Product Lifecycle Management (PLM).

This will be achieved by creating a system that transfers this information by the means of XML web services. This thesis focuses on the architecture on the TERPS side. The system on the TERPS side will automatically receive information from Teamcenter through the WESB platform. The information will be inserted into a staging database where it waits for further approval. From there it is finally inserted into the endpoint database, using an ASP.NET web application.

The ASP.NET web application is a tool for presenting and approving information in the staging database and updating the TERPS database.

The result is a solution that transfers information from Teamcenter to the TERPS database.

Language: english Key words: Teamcenter, TERPS, web service

EXAMENSARBETE

Författare:	Niklas Järnström
Utbildningsprogram:	Informationsteknik, Vasa
Handledare:	Ray Pörn

Titel: *Produktdataöverföring mellan globala och lokala system*

Datum: 21.3.2013 Sidantal: 34 Bilagor: 3

Abstrakt

Detta examensarbete gjordes på uppdrag av Wärtsilä Finland Oy, Ship Power, och handlar om produktdataöverföring från en global applikation, Teamcenter, till en lokal databas, TERPS. Syftet är att expandera Product Data Management (PDM) för att även täcka projektleveranser eller Product Lifecycle Management (PLM).

Detta åstadkoms genom att skapa ett system som använder sig av XML-webbtjänster för att överföra information. Detta examensarbete fokuserar på arkitekturen på TERPS- sidan. Systemet kommer automatiskt att ta emot information från Teamcenter genom WESB-plattformen. Denna information kommer att placeras i en mellanlagringsdatabas där den stannar tills den blivit godkänd genom att använda en ASP.NET-webbapplikation.

ASP.NET-webbapplikationen är ett verktyg för att presentera och godkänna information i mellanlagringsdatabasen och uppdatera TERPS-databasen.

Resultatet är en lösning som kan flytta information mellan Teamcenter och TERPS databasen.

Språk: engelska Nyckelord: Teamcenter, TERPS, webbservice

OPINNÄYTETYÖ

Tekijä: Niklas Järnström

Koulutusohjelma: Informaatiotekniikka, Vaasa

Ohjaaja: Ray Pörn

Nimike: *Tuotetiedon siirto globaalin ja paikallisen systeemien välillä*

Päivämäärä: 21.03.2013 Sivumäärä: 34 Liitteet: 3

Tiivistelmä

Tämän opinnäytetyön toimeksiantaja on Wärtsilä Finland Oy Ship Power. Työ käsittelee tuotetiedon siirtämistä, globaalista applikaatiosta Teamcenter paikalliseen tietokantaan TERPS. Tarkoitus on laajentaa Product Data Managament (PDM) käsittämään myös projektitoimituksia tai Product Lifecycle Managementia (PLM).

Tämä toteutuu XML-verkkopalvelun avulla. Verkkopalvelu siirtää tietoa käyttämällä Xml:ää. Opinnäytetyö keskittyy TERPSin puolen arkkitehtuuriin.

Systeemi saa automaattisesti tietoa Teamcenterista WESB-alustan kautta. Tieto sijoitetaan välitietokantaan, missä se pysyy kunnes se on hyväksytty käyttämällä ASP.NET-verkkosovellusta.

ASP.NET-verkkosovellus on työkalu, jolla esitetään ja hyväksytään välitietokannan tietoa ja päivitetään TERPS-tietokantaan.

Tulos on sovellus, joka siirtää tietoa Teamcenterin ja TERPS-tietokannan välillä.

Kieli: englanti Avainsanat: Teamcenter, TERPS, Web Service

Table of Contents

Abbreviations

ASMX	Active Server Methods, a Microsoft Technology
ASP	Active Server Pages, a Microsoft Technology
HTML	Hyper Text Markup Language
IDM	Integrated Document Management
IPI	Installation Planning Instructions
IPIX	Information Publishing in XML
JSON	Java Script Object Notation
PDF	Portable Document Format by Adobe
PDM	Product Data Management
PLCS	Product Life Cycle Support
PLM	Product Lifecycle Management
PLMXML	Product Lifecycle Management XML, a Siemens format for facilitating product lifecycle interoperability using XML
QA	Quality Assurance
QMS	Quotation Management System
SOAP	Simple Object Access Protocol
TERPS	Total Engine Room Packages for Auxiliary Systems
WCF	Windows Communication Foundation
WDMS	Wärtsilä Data Management System
WIO	Wärtsilä Industrial Operations
WS	Web Service
WSDL	Web Service Description Language

ACKNOWLEDGEMENTS

I would like to express my gratitude to: Mr Leif Backlund and Mr Robert Tuure, who have contributed to the creation of this project.

1 Introduction

This thesis concerns a project commissioned by Wärtsilä Finland – Ship Power and is about transferring product data between global and local systems. Wärtsilä is in progress of expanding the Product Data Management (PDM) to cover also project deliveries or Product Lifecycle Management (PLM).

PDM is currently used mostly for capturing product design data. A software called Teamcenter will be used for this phase but Wärtsilä still has a need to transfer product data into local databases to improve speed and flexibility.

In order to achieve this a system that handles the transfer of product data between the two systems has to be designed, built and implemented.

This thesis focuses on the transfer of product data from Teamcenter software to a local database called TERPS (Total Engine Room Packages for Auxiliary Systems). The content definition is based on the Product Life Cycle Support (PLCS) which is an ISO STEP standard (ISO 10303-239).

The purpose of this thesis is to produce enough material for being able to design a specification for a production quality assurance environment (QA).

2 The assignment

My thesis work is a project concerning data transfer between global and local systems. This means utilizing product data from Teamcenter software into a local database called TERPS (Total Engine Room Packages for auxiliary Systems).

The project is split into several parts which will be handled by different parties. This thesis focuses on the product data transfer between Teamcenter and the TERPS database. The database is a local database containing information about auxiliary equipment, e.g. description, type, technical specifications and prices, etc. The database is used by several other systems and applications, e.g. the configurators and IPI.

Currently the database has its own interface for updating information about components. The information about these components has to be stored and handled by a global system called Teamcenter. In the future Teamcenter will be the master of this information, but in order to improve speed and flexibility, the information still needs to be stored in the TERPS database.

The focus lies on transferring the data from Teamcenter into the TERPS database. This means studying the existing systems and their usage in order to get a clear overview on how the problems should be solved. The next step is to study and present different alternatives of how the transfer should be done and make a feasibility study of these different alternatives, e.g. functionality , design etc. When a suitable solution has been found, a basic design of that solution is to be made. Following the final specifications, rules and documents, the solution should be verified with some possible test data.

The thesis will start with a short overview of the TERPS database and its related systems, as it was at the start of the project. The overview will be followed by a requirements specification part and a deeper look into how different issues were solved and how the solution was implemented. Finally there is a discussion part.

3 TERPS

The TERPS database is a Microsoft Access 2007 database. The database consists of information about auxiliary equipment (also referred to as components) which is used in Wärtsilä 4-stroke engine applications. The name TERPS was invented in the early 1990's, since there was a need to collect information about auxiliary equipment in one place. This was important since the approved information is sent to customers. The database has been modified several times since then, from the earliest PDF versions to today's relational database. /1/

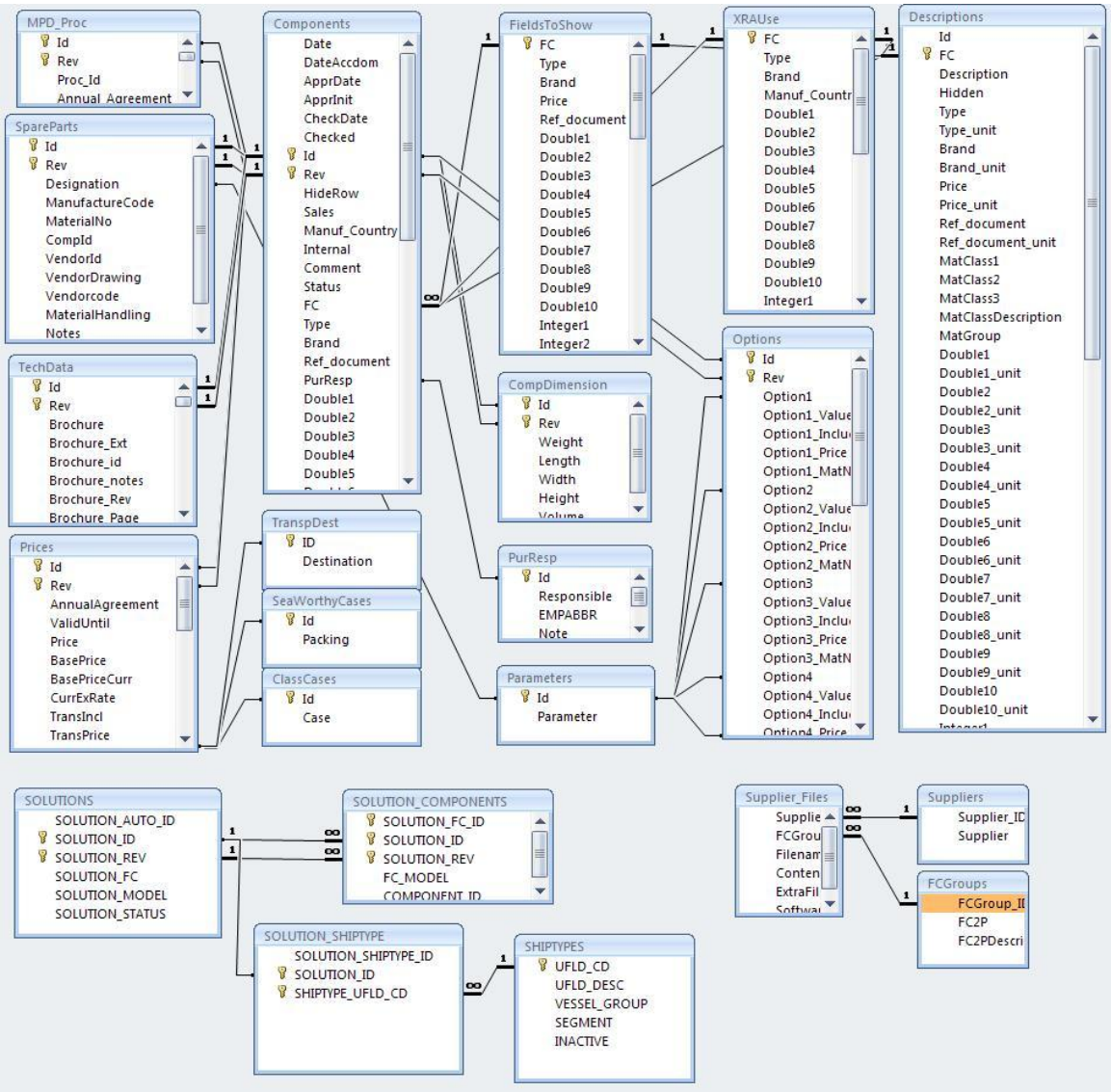


Figure 1. TERPS Relationships

Figure 1 shows the database diagram of the TERPS database, prior to any of the modifications made that were necessary for this thesis.

3.1 TERPS usage

The TERPS database is used by a number of other systems and applications, such as the configurators, IPI, TERPS Interface and the TERPS Admin interface. These applications and systems will be more thoroughly described in the following chapters. The usage of the database by other systems and applications is visualized in the picture below.

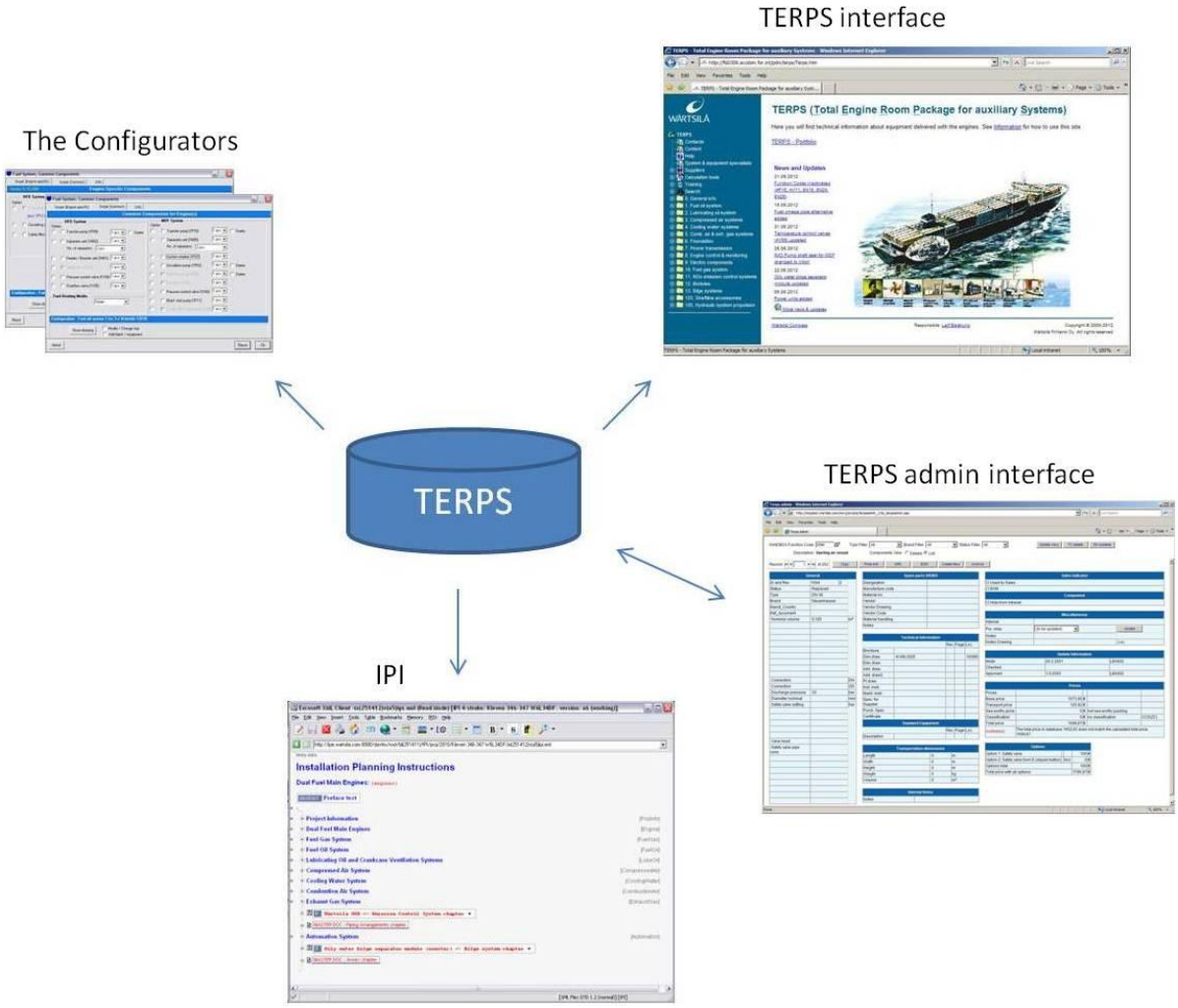


Figure 2. Visualization of TERPS usage

3.1.1 The configurators

The configurators form a part of the Quotation Management System (QMS). The configurators are a tool used when making a quotation for a project. When making a quotation on a certain component, the configurators select data from the TERPS database, based on formulas and rules inside the configurators. The configurators then release the information to the QMS database.

3.1.2 Installation Planning Instructions (IPI)

Wärtsilä as a supplier and the yard as a contractor need to communicate installation specific information to build an installation that is both reliable and economical to use. The purpose of the IPI is to support Wärtsilä customers and Wärtsilä network offices by giving them a proper picture of the complete scope of supply from Wärtsilä. Based on the information in IPI the customer will be able to integrate Wärtsilä scope of supply to the rest of the ship engine room and propulsion system. /11/

The IPI consists of:

- Standard scope of supply and related systems
- Recommendations, requirements and design data
- Project specific drawings and technical data (retrieved from TERPS).

3.1.3 TERPS interface

The TERPS interface is a way of viewing the content of the database. The information is visualized on ASP classical pages. Through the interface a user can easily browse the components in the database, either by selecting the desired component in a certain component group or by just searching for the component using a specific search criterion. Component specific data, e.g. prices, technical specifications, drawings and brochures (stored in WDMS), are very easy to browse.

The TERPS interface also stores other information and contents such as:

- System / Equipment specialists
- Documents (stored in WDMS)

- Design formulas (stored in WDMS)
- Supplier software and files.

The interface is built using ASP Classical, HTML and JavaScript.

3.1.4 TERPS administrator interface

The administrator interface is also built with ASP, HTML and JavaScript technique. It allows certain users (administrators) to e.g. update, check status and create new components in the database, i.e. maintain the database. The administrator interface also shows the data from earlier revisions of the components, in other words it shows components whose data have changed and are no longer under production status. Thus, the TERPS administrator interface is a “maintenance tool” for the TERPS database.

4 Teamcenter

Teamcenter is an integrated suite of Product Lifecycle Management applications, originally created by Siemens PLM Software. Teamcenter connects people and processes by creating, capturing and sharing product knowledge to power innovation and productivity. Teamcenter is the world's most widely used PLM system. Teamcenter product capabilities are for example: Bill of Materials Management, Content and Document Management, Lifecycle Visualization and more. Today Wärtsilä is in the progress of implementing Teamcenter into their Product Document Management. The implementation of Teamcenter is split up into several phases and is currently in the Beta phase. /10/

Although the retrieval of data from Teamcenter lies outside the scope of the thesis (it will be handled by another party), it still serves as the key building block. The goal is that Teamcenter becomes the master of the TERPS database content and all changes will be made using Teamcenter./10/

5 Requirements

As was mentioned earlier, to be able to expand the PDM to cover also project deliveries, global application data has to be transferred into local databases (TERPS). Several meetings were held at Wärtsilä, with the purpose of discussing the requirements and the functionality needed by the data transferring solution. The figure below shows the TERPS - Teamcenter integration as it was at the beginning of the project.

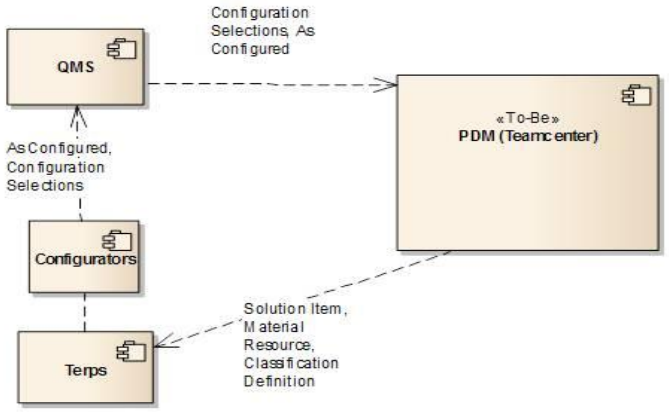


Figure 3. TERPS integration

During the meetings, different alternatives of tackling the problems that arose were discussed. During the early stages of the thesis work I proposed a solution where the transferring of data would be done through a web service that uses XML to transport data. This solution seemed to be very suitable from a Wärtsilä point of view, because something similar had already been done.

Furthermore, a suggestion was made that the existing TERPS database should be converted from an MS Access 2007 database into a SQL Server database. The conversion into SQL Server would be a far better alternative than MS Access, since SQL Server can handle the load of simultaneously accessing users much better than MS Access 2007 /7/. According to Wärtsilä internal statistics, the usage of the TERPS interface has increased a lot in the past few years and the interface uses the database as data source, which in turn results in increased access to the database. The database was likely to be reconstructed in order to be able to map the transferred data (global) with the locally existing data, e.g. by adding new columns to associate the data. The idea was to do this at the same time, thus minimizing the time span needed for database changes.

Indeed, a database conversion from MS Access to SQL Server would mean changes to all other systems using the TERPS database, for example the configurators and the TERPS interface. During one of the meetings we concluded that the changes required by the database conversion should be easy to fix in a relatively short time.

The focus of this thesis is on the creation of a system that handles XML data. Data from Teamcenter will be sent to this system through an integration platform (WESB). This process is handled by another party. The scope of this system is to create a web service (WS) that receives data from WESB and an ASP.NET web application that presents the data received by the web service. Figure 4 shows the proposed future architecture of TERPS (with thesis focus part highlighted in red) and figure 5 shows the existing architecture.

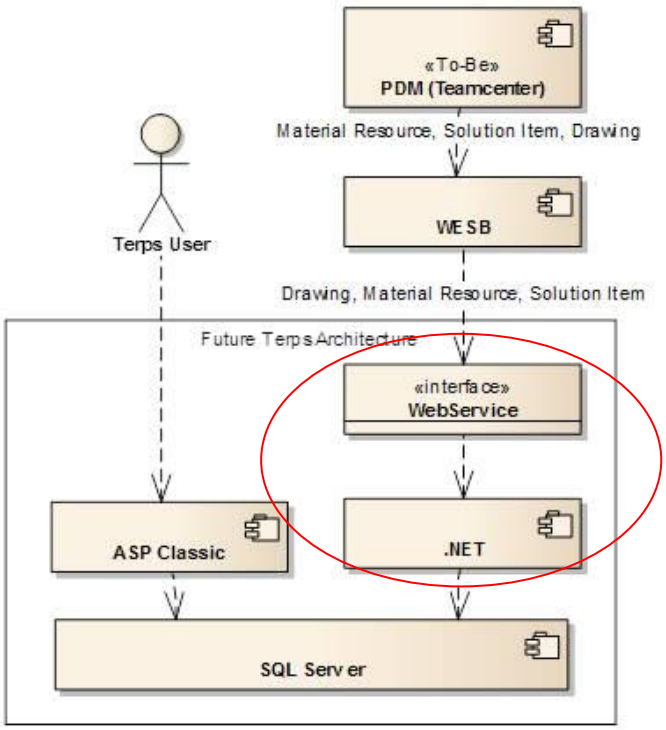


Figure 4. Proposed TERPS architecture

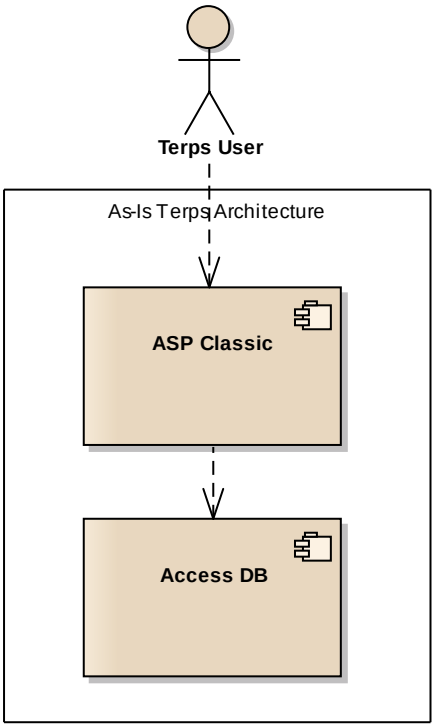


Figure 5. TERPS side architecture

As seen above in figure 4 the preferred solution will use a web service with web callable methods. This web service will receive transferred data. This data will then be presented by an ASP.NET web application. The web service itself will further on be invoked from WESB.

The web service, which receives data from Teamcenter, needed to have a certain structure in order to be able to receive data about solution items and send back response messages about these items. This was achieved using a technique called “contract first”. This means creating a service stub from a pre-existing metadata document (a WSDL file).

The application presenting the data was determined to be an ASP.NET web application in order to allow easy accessibility. The purpose of this application is to act as a tool for approving the updates in the endpoint database (TERPS). This will be done by presenting the data from both systems (Teamcenter and TERPS) in a tabular format. This allows any user to easily interpret the data that is to be updated.

6 The solution

This thesis is part of a greater solution covering the data transfer of solution items from product lifecycle management system Teamcenter to TERPS database. To transfer a solution item from Teamcenter to TERPS, a number of data transformations have to be done. The flow of the data is presented in figure 6 below, where WS represents the web service on the TERPS side.

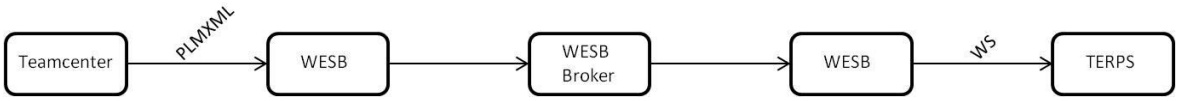


Figure 6. Solution item flow

Solution item transfer	
Description	Solution item transfer from Teamcenter to TERPS
Goal	To transfer information about solution items including classifications, material recourses and other attributes related to auxiliary equipment.
Summary	The solution item data process integration provides a way to transfer solution items from Teamcenter system to the consuming system TERPS, transmitting this information trough WESB.
Actors	• Teamcenter • WESB (“Teamcenter adapter”) • WESB (“TERPS adapter”) • TERPS
Trigger	The message is triggered from Teamcenter system workflows (event based)
Steps	1. A PLMXML message containing solution items is received from Teamcenter through a SOAP call.

	2. The data is mapped to a canonical structure by WESB 3. WESB publishes the data to the Broker 4. The data is dequeued by WESB (TERPS adapter) 5. WESB calls TERPS web service 6. WESB presents transfer status to vision
Alternatives	The PLMXML message received from Teamcenter is invalid. This causes the process to terminate and an error message is sent to Vision and Teamcenter.

Table 1. Solution item transfer from Teamcenter to TERPS

Table 1 gives a more accurate description of the several steps taken before the solution items are updated in the end point database. As can be seen in the table, WESB logs a transfer status of the solution items in order to notify Teamcenter of a successful update in TERPS. The transfer status can also be viewed using the Vision application. This means that the TERPS web service needs to send a response message about the transfer status. The flow of the response is presented in figure 7.

Figure 7. Response message flow

When a solution item is released in Teamcenter, the information about this particular solution item is sent to TERPS web service according to the material presented in the previous paragraph. If this information is successfully handled on the TERPS side, a response message is sent back to Teamcenter telling that the transfer was carried out successfully. If an error occurs the message will contain information about the cause of the error. The flow of the response message is presented below in table 2.

Response message transfer	
Description	Response message transfer from TERPS to Teamcenter
Goal	To transfer a response message regarding the status of a particular solution item in TERPS to Teamcenter.
Summary	The status process integration provides a way to transfer status regarding the status of solution items from the system to the consuming system TERPS to Teamcenter, transmitting this information through WESB
Actors	• TERPS • WESB (“TERPS adapter”) • WESB (“Teamcenter adapter”) • Teamcenter
Trigger	The message is triggered from Teamcenter system workflows (event based)
Steps	1. An XML message containing the status is received through a SOAP call 2. The data is mapped to a canonical structure by WESB 3. WESB publishes the data to WESB Broker 4. The data is dequeued by WESB (Teamcenter adapter) 5. WESB calls Teamcenter web service 6. WESB presents status to vision
Alternatives	The response message sent from TERPS is invalid. This causes the process to terminate and an error message is sent to Vision.

Table 2. Response message transfer from TERPS to Teamcenter

This chapter has given a description of the whole solution design, although it partly lay outside the scope of this thesis. The following chapter will present a closer look at the

TERPS web service, the .NET web application presenting the solution items and finally the staging database.

6.1 TERPS side architecture

The scope of this thesis lies in creating a web service that receives solution items from another web service. This web service will then handle the data and update the local TERPS database. The web service does not directly update the local TERPS database, instead it uses a staging database. By Wärtsilä request a .NET web application displaying the data in a tabular format was also created. This will present the data in an interpretable format for any user.

6.2 Planning the web service

Several meetings were held at Wärtsilä with the purpose of discussing how the web service should be created, i.e. which technique should be used. The first idea was to create an ASP.NET web service application. This seemed like a good idea from a Wärtsilä point of view, but after a workshop held with one of the persons involved in the WESB part we were advised to create a WCF service instead.

Although an ASP.NET web service application (.asmx) could work fine, WCF is still a better choice, since it supports all the capabilities that ASMX does. In addition, WCF allows more than ASMX, e.g. Message Structure and Serialization, SOAP Extensions and additional transport protocols. /4/

With the information stated in the previous paragraph and the advice from one of the persons involved with WESB, a WCF service was created and tested. The problem that arose with this option was that WCF exposes metadata (WSDL files) as several files. This means that there is a WSDL file in which there is a link to external a XML Schema. This was a problem, because in order to invoke this service from WESB, it needed to present its data as a flat WSDL file (no external files).

However, this may be solved if you explicitly tell the WCF service to expose its metadata as a single file, using the query string “?singleWSDL”./2/ Unfortunately, this is only possible if you use .NET 4.5. There were several other things that prevented me from using it. For example, .NET 4.5 is not supported in Microsoft Windows XP. These limitations

forced me to abandon the idea of creating a WCF service. Instead an ASP.NET web service application (ASMX) was created, as originally planned.

6.2.1 SOAP (Simple Object Access Protocol)

An ASP.NET web service application uses SOAP as transport protocol. Thanks to that a consumer of this service does not need to know what platform or programming language is used to implement this service. A consumer only has to understand how to send SOAP messages (which is basically HTTP and XML). Figures 8 and 9 show example SOAP request and response messages, used by the service, where the **placeholders** shown are replaced with real values. /5/

```
POST /Service1.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "pdm_terps_SolutionItemService_Binder_postSolutionItem"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <postSolutionItem xmlns="http://*****/pdm/terps/SolutionItemService">
      <SolutionItem xmlns="">
        <RequestHeader>
          <workFlowId>string</workFlowId>
          <workFlowName>string</workFlowName>
        </RequestHeader>
        <ProductData>
          <RootItem itemId="string" description="string"
globalLifeCyclePhaseCode="string" globalProductTypeCode="string"
globalProductUnitOfMeasureCode="string" revisionId="string">
            <ICO xsi:nil="true" />
            <AdditionalAttributes xsi:nil="true" />
            <File xsi:nil="true" />
            <MaterialResources xsi:nil="true" />
            <BillOfMaterial xsi:nil="true" />
            <Document xsi:nil="true" />
            <Cost xsi:nil="true" />
          </RootItem>
          <SubItems>
            <SubItem xsi:nil="true" />
          </SubItems>
        </ProductData>
      </SolutionItem>
    </postSolutionItem>
  </soap:Body>
</soap:Envelope>
```

Figure 8. Complete example of the SOAP Request

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <postSolutionItemResponse xmlns="http://**/pdm/terps/SolutionItemService">
      <Response xmlns="">
        <Response>
          <Header xmlns="http://**/pdm/terps/SolutionItemService">
            <trackingID>string</trackingID>
            <trackingReference xmlns="">string</trackingReference>
            <timeStamp>string</timeStamp>
            <sender>string</sender>
            <receiver>string</receiver>
          </Header>
          <status xmlns="http://** /pdm/terps/SolutionItemService">string</status>
          <message xmlns="http://**/pdm/terps/SolutionItemService">string</message>
          <AdditonalParameters>
            <AdditionalParameter xsi:nil="true" />
          </AdditonalParameters>
          <Error xmlns="http://**/pdm/terps/SolutionItemService">
            <errorMessage>string</errorMessage>
            <errorCode>string</errorCode>
            <errorType>string</errorType>
            <service>string</service>
            <reference>string</reference>
          </Error>
        </Response>
      </Response>
    </postSolutionItemResponse>
  </soap:Body>
</soap:Envelope>
```

Figure 9. Complete example of the SOAP Response

6.2.2 The solution item service

The web service is an ASP.NET web service application and is coded using C# programming language. The service implements a single service interface IPdm_terps_SolutionItemservice_Binder, which has a method postSolutionItems for retrieving information about the solution items. The classes for the solution items were also needed. This interface and the classes were created from a WSDL file provided by the party involved in the WESB platform. The WSDL file provided served as a starting document. The file was then changed according to the structured data being sent to the service. The WSDL file or the meta-data document consists of information about what methods the service should implement (in form of XML) and an XSD schema telling how the classes need to be constructed. The interface and classes were auto generated using Microsoft SDK tool wsdl.exe. The technique of creating web services from pre-existing metadata documents (WSDL files) is called “contract first”.

```
[System.CodeDom.Compiler.GeneratedCodeAttribute("wsdl", "2.0.50727.3038")]
[System.Web.Services.WebServiceBindingAttribute(Name="pdm_terps_SolutionItemService
_Binder", Namespace="http://***************/pdm/terps/SolutionItemService")]
public interface IPdm_terps_SolutionItemService_Binder {

    /// <remarks/>
    [System.Web.Services.WebMethodAttribute()]

[System.Web.Services.Protocols.SoapDocumentMethodAttribute("pdm_terps_SolutionItemS
ervice_Binder_postSolutionItem",
RequestNamespace="http://*****************/pdm/terps/SolutionItemService",
ResponseNamespace="http://***************/pdm/terps/SolutionItemService",
Use=System.Web.Services.Description.SoapBindingUse.Literal,
ParameterStyle=System.Web.Services.Protocols.SoapParameterStyle.Wrapped)]
    [return: System.Xml.Serialization.XmlElementAttribute("Response",
Form=System.Xml.Schema.XmlSchemaForm.Unqualified, IsNullable=true)]
    GenericResponse
postSolutionItem([System.Xml.Serialization.XmlElementAttribute(Form=System.Xml.Sche
ma.XmlSchemaForm.Unqualified, IsNullable=true)] SolutionItem SolutionItem);
}
```

Figure 10. The interface that the service implements

The service implements a single method postSolutionItem from the interface generated by wsdl.exe (presented in the code snippet in figure 10). This method takes an object of class SolutionItem as input parameter and returns a response in form of an object of the GenericResponse class. The classes are provided by the service contract and generated by wsdl.exe. When a solution item is received, the service attempts to open the connection to a staging database and inserts the information about the received solution item. If this operation is successful, it will return a response message containing information about a successful transfer for the specific solution item. Note that this is only a message telling that the information was successfully inserted in the staging database, *not* the end point database (TERPS). If the insertion fails for some reason, the service will do a rollback and return a response message containing errors. The response message and the corresponding data can be viewed using the Vision application (appendices I and II). The following figures 11 and 12 show sample data of a received solution item and the corresponding response message returned by the service. Note that the information is only fictive, but it is comparable to real data.

Figure 11. Solution item data example

Figure 12. Response message of corresponding solution item

The decision to make the web service insert data into a staging database instead of directly into the TERPS database comes from the ASP.NET web application. The purpose of this application is to present the information in a format that any user can interpret, not only in XML format. In other words, this application will serve as an update tool. The transfer of solution items from Teamcenter to the TERPS web service is nearly real time. The ASP.NET web application is a *manual* update tool and the web service is then forced to save information about solution items in a staging database until it is approved, using this manual update tool. This means that the staging database is a blank one to one copy of the TERPS database. The only purpose of the staging database is to gather information about solution items that are going to be updated using the ASP.NET web application.

The staging database is a one to one copy of the end point database (TERPS) with the addition of stored procedures used by the web service to insert items. The staging database is empty until a solution item is inserted by the web service. That solution item is then retrieved by the ASP.NET web application and inserted in the end point database (TERPS) and then deleted from the staging database. This means that the staging database will only hold information about solution items that have not yet been updated in TERPS, i.e. information that is under the process of being transferred.

The main tasks of the web service interacting with WESB are as follows:

- Receive solution item from WESB
- Process received solution item
- Insert solution item into staging database
- Send response to WESB about successful or unsuccessful insert into staging database

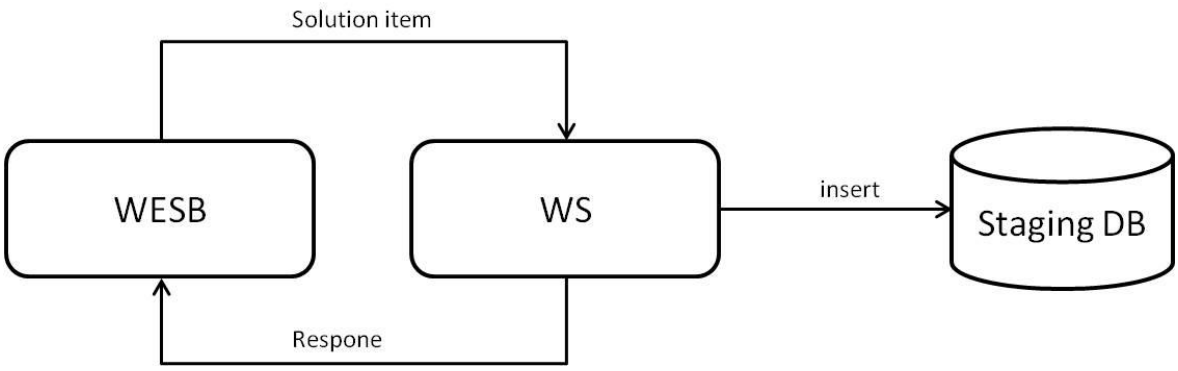


Figure 13. Visualization of TERPS side web service

Figure 13 is a visualization of how the process works after a solution item is received from WESB until it is inserted in the staging database.

6.3 TERPS Update tool

6.3.1 Structure and functionality

The TERPS update tool is a an ASP.NET web application, with the code behind files written in C#. As mentioned earlier the purpose of this application is mainly to present data that is to be transferred from Teamcenter to TERPS.

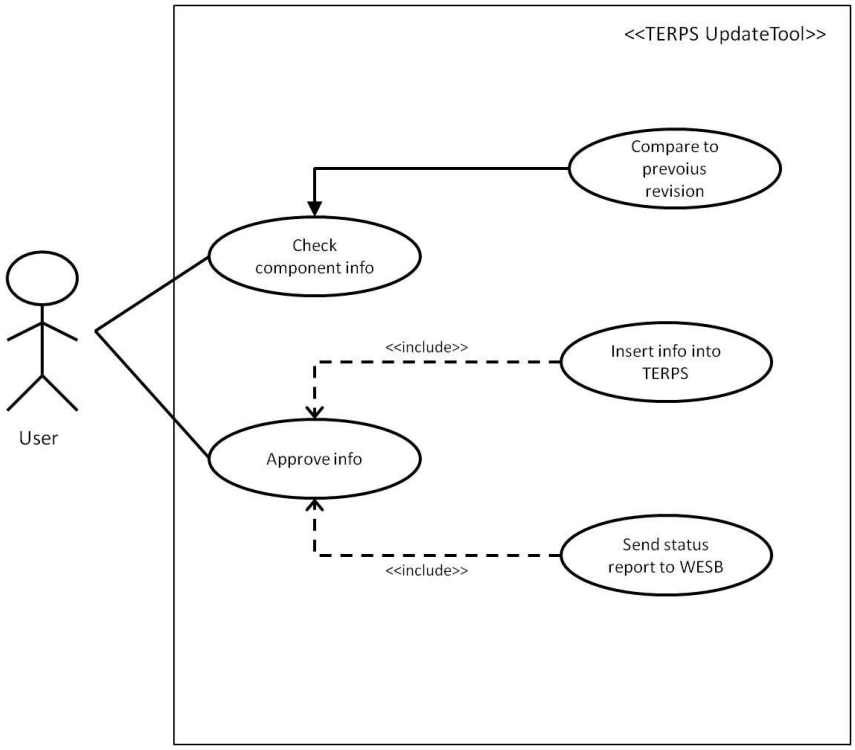


Figure 14. Simplified version of the use case diagram

In figure 14 it can be seen that a user can:

- Check the information about items being transferred (from staging database), and compare them to the items' previous revisions (from endpoint database).
- Approve the info which includes inserting the new info into TERPS database and sending a status report to WESB web service StatusReportProvider

The tool simply presents the data from the staging database (contains information about components under the process of being transferred) in a tabular format. If a user then clicks

on a certain component, the user will be redirected to another page that lists the information about the component in question and the corresponding component in TERPS. For example, if a component with id XAA123 and revision 2 has been created in Teamcenter, this component would be compared to a component in TERPS with id XAA123 and revision 1, which is the component with the highest revision. If a new id number is transferred from Teamcenter, no comparison can be made because there is no matching id that can be compared to the transferred id.

In order to make the information orderly, the first page only presents data in the staging database. By clicking on tabs, the user can change the table, from which the application shows data. By clicking a component, the user is redirected to another page where the component information is compared to the component with the highest revision.

Figure 15 below is a basic sketch and visualization of the things explained in the previous paragraph. The data is fictive, but comparable to actual data.

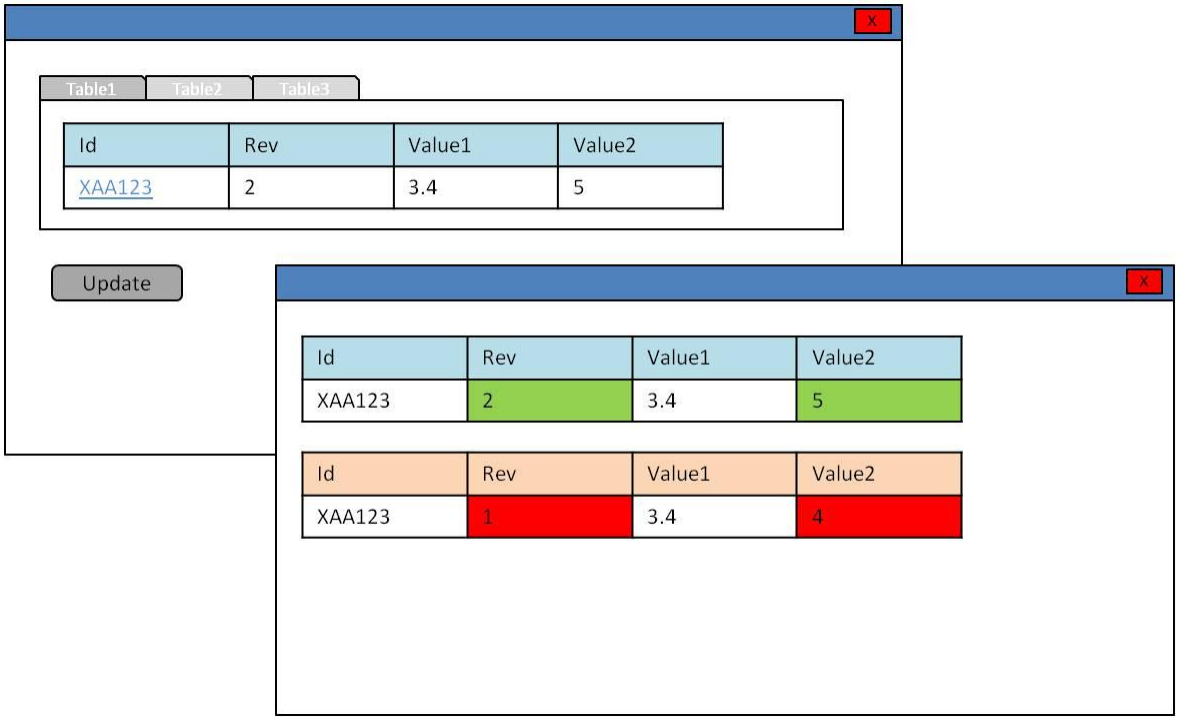


Figure 15. Visualization and basic sketch of TERPS update tool

If a user checks this information and finds it satisfactory, the user can simply approve the insertion of this information into TERPS by simply clicking an update button. The application will then start a process of transferring these solution items one by one, using a

stored procedure in the staging database. If a solution item is successfully transferred, it will be deleted from the staging database. In order to improve security, this process is handled by SQL transactions. This means that the whole process has to succeed (not only a single item but all items). If the process doesn't succeed, neither the insert into TERPS nor the deletion from the staging database are executed. In the case of failure, the user will also be provided with the reasons and the item that caused this failure.

6.3.2 Response messages

Included in the transfer process is also a call to the WESB web service "StatusReportProvider" which informs about a successful insert in the final database. This call is necessary in order to tell Teamcenter that the workflow for the solution item in question can be released. This call deviates from the response message mentioned in the previous chapter in only two ways. Either it informs about a successful or an unsuccessful transfer through the whole chain into the TERPS database, i.e. not only a successful insert in the staging database (as the previous message does) and it also informs Teamcenter if the inserted item is under production or test status. Figure 16 shows both of the response messages sent back to Teamcenter. The response message can also be viewed graphically using Vision application (appendix III).

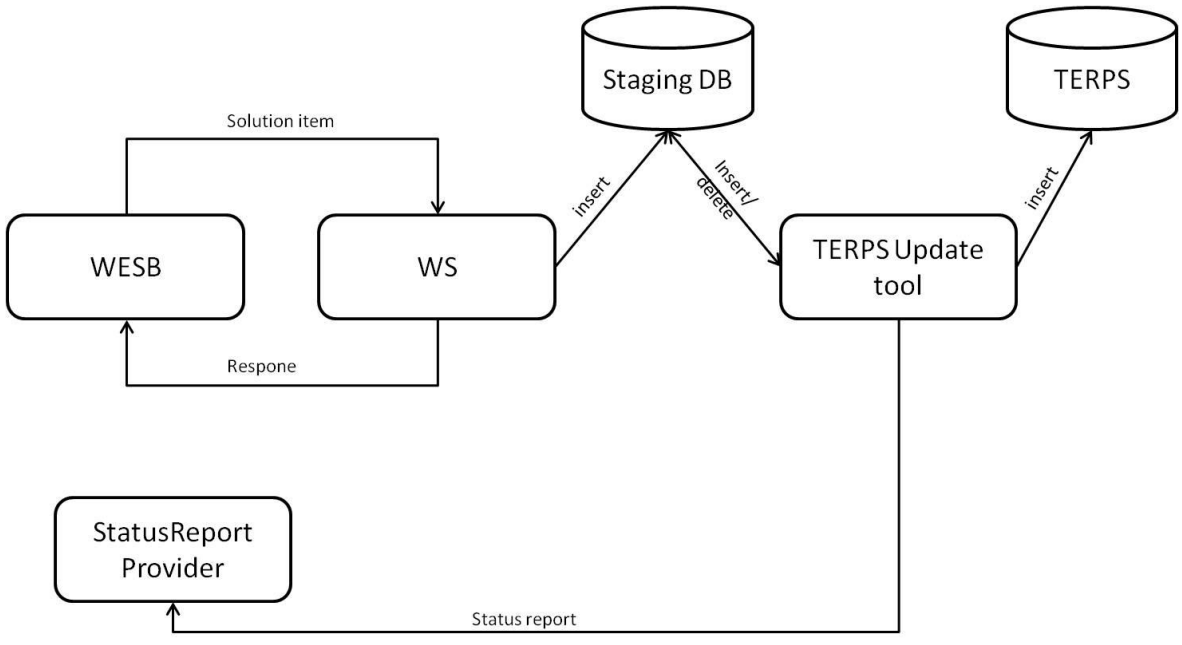


Figure 16. Update tool response message flow

6.3.3 Tasks of the ASP.NET web application

As mentioned at the beginning of this chapter, the purpose of the update tool is only to present the data in a way that any user can understand. The tasks performed by the tool can be simplified to:

- Check for and display information in staging database
- Update TERPS data base
- Send status report about updated solution items

Based on this information one can say that the application is just a check for the human eye on what is going to be inserted into the endpoint database. Since the process of transferring a solution item from Teamcenter to the staging database is near real time, it raises the question if this application is needed at all.

The answer to the question lies in the configurators that use the information in the TERPS database. Before approving information transmitted from Teamcenter, it has to be verified that this information works with the configurators. By using the staging database and the ASP.NET web application, there is theoretically no time limit on the final update in TERPS database after an item has been created in Teamcenter. The information simply stays in the staging database until it is transferred to TERPS by the update tool.

In addition to the ASP.NET web application itself, a manual was written. The manual describes the functionality of the application and also explains how to modify the SQL stored procedures in case of future database changes.

6.4 The TERPS database

Included in the TERPS side architecture are also the two databases TERPS and TERPS_staging. Both data bases are SQL Server 2008 databases originally created from the Microsoft Access 2007 database.

The conversion from Microsoft Access 2007 into SQL Server 2008 comes with both advantages and disadvantages. For example, an SQL Server 2008 allows more complex stored procedures to be created in the database. Stored procedures form one of the key elements in this particular solution, because they make the application code more independent of changes in the database structure. /8/

There are also disadvantages. For example, any existing application that reads from the database, such as the TERPS interface (ASP pages) will need to be changed in order to function properly with the converted database. However, comparing the advantages with the disadvantages, it is clear that SQL Server 2008 is a far better alternative than Microsoft Access 2007.

6.5 The TERPS staging database

The TERPS_staging database functions as a middle storage database for solution items that are under the process of being transferred. A staging database can function in several different ways:

- It can gather information about components that have been transferred. This means that the database will always contain information about items that have been transferred. The positive thing about this way is the ability to check the history of transferred items. The negative thing is the continuous growth of the database.
- It can gather information about components that are under the process of being transferred. This means that the database will only contain information about items not yet transferred to the end point database. When all items are transferred, the database will be blank. The positive thing about this alternative is that the database will not grow in size over time and it is easier to select the information that is going to be transferred. On the other hand, the disadvantage is the inability to check the history of transferred items.

In this case the second alternative was chosen, since the history of transferred items is logged by another application, namely Vision . The data of the staging database can by used in many ways, for example: to find changes against the current data warehouse (TERPS endpoint database), data cleansing or to pre-calculate aggregates. In this solution the data from the staging database is compared to the existing data in TERPS, allowing an easy find of changes. A staging database can also gather information from many different sources, but in this case it only gathers information from one source. The information is then to be updated later on by using the ASP.NET update tool. /9/

6.6 Summary of the architecture on the TERPS side

The solution works fine but is rather complex, with many different parts (web services and applications), all playing their own roles in the transfer of information. The complex structure can be negative in case of future changes. A worst case scenario would be to have to rewrite code for every service and application involved in the process of transferring solution item information. The amount of work this would demand can be multiplied by three, since the system is located on test, development and production servers. However, the software on the TERPS side is designed to avoid worst case scenarios, like the one described.

When summarizing the main components on the TERPS side required to achieve the goals specified in the Requirements chapter, they can be presented as three major parts:

- TERPS SolutionItem web service
- TERPS_staging database
- TERPS Update tool

All these components have their own roles in the transfer of a solution item. In fact there are a lot more steps taken before the information has reached the TERPS side. These steps have been briefly described in chapter 5 and are not going to be more thoroughly described because they lie outside the scope of the TERPS side architecture.

7 Security and data quality

7.1 Data quality

During this project we always had to keep in mind that the quality of the data is of great importance. One major requirement for this thesis was to be able to ensure that the quality of the data stays the same through the whole transfer from Teamcenter to TERPS.

The word data quality means that the data needs to correctly represent the real-world construct to which they refer. According to J.M. Juran, data is deemed to be of high quality "if they are fit for their intended uses in operations, decision making and planning". If the quality in Teamcenter is considered to be high, the system transferring the data can't do any changes to the data which will affect the quality./3/

The TERPS database is consumed by a number of other applications and systems (e.g. IPI and the configurators). These systems produce data based on the data retrieved from TERPS. The data produced by the consuming systems is also sent to customers. In other words, the quality of the data in the TERPS database is of great importance and is directly linked to the system transferring the information from Teamcenter.

In order to ensure the quality of the data being transferred, the applications on the TERPS side provide no user inputs at all. This is to minimize the risk of human errors. If a user for example uses the TERPS update tool and is allowed to change the data being transferred, there will be a mismatch of the databases because the transmitted and received information is *not* the same. To totally eliminate the risk of human errors is impossible, because the data itself is created by a human being.

7.2 Security

Based on the information presented in chapter 7.1, a decision was made to allow no user inputs at all in the ASP.NET web application. This makes the creator of the data in Teamcenter responsible for the data quality. The application would then just be as a check for the human eye, telling what information is being transferred. It would also give a possibility to verify that the information works with other systems, consuming the TERPS database, before it is approved.

Even though the update tool (ASP.NET web application) allows no user inputs or data modifications, it greatly improves the security of the transfer process as it serves as an additional opportunity to check the data sent from Teamcenter.

Possible errors can then be found before the incorrect information is inserted into the TERPS database (which would be critical).

8 Alternatives to the architecture on the TERPS side

8.1 Making the system adaptable to changes

While working on this project it was important to note that the system has to be adaptable to database structure changes. For example, if a new column is added in the TERPS database, it cannot have critical effects on the system transferring the data from Teamcenter to TERPS. Since modifications have been made before, the question is not: *if* the database structure changes, but *when* the database structure changes

This has been achieved (as far as possible) by writing code that e.g. is not dependent of the database structure. The application and web service use SQL stored procedures as far as possible when retrieving or inserting data from or into a database. This means that if one of the databases changes in some way, the code for the application or web service does not need to be rewritten and recompiled. Instead only an alteration of the existing procedures is required. Of course, a solution that is totally independent of database changes may be impossible to create.

The web service receiving the information about solution items from Teamcenter was the hardest thing to make independent of changes. During the project it was proposed that the web service would insert the data as XML data directly into the database. This would have meant that the staging database would no longer be a one to one copy of the endpoint database. Instead it would only contain fields with xml as data type. The ASP.NET web application would then read these XML files and insert information into the TERPS database. This solution would make the web service totally independent of what information is transferred. The only thing it would do is insert XML data into the staging database.

This solution sounded as a very interesting solution and it was discussed with the party involved in WESB. It was partly implemented in order to test it, but unfortunately it was presented too late to be able to finish testing it within the time limitations of this thesis.

8.2 XML versus JSON

There is also the alternative of using JSON instead of SOAP. Both JSON and XML (used by SOAP) are text based, which gives them the same interoperability. However, JSON is considered to be much simpler than XML. In addition JSON is considered to be much easier to process due to simpler structure. JSON is also easier for mapping data object-oriented systems because JSON itself is data-oriented (compared to XML which is document-oriented). /6/

However, the choice of sending the information as XML instead of JSON comes from the PLMXML. The information is retrieved from Teamcenter as a PLMXML (used by Siemens for facilitating product lifecycle interoperability using XML) message through a SOAP call. It would then be rather unfitting to convert the information to JSON.

Using JSON in this particular case would also mean a lot of extra work. Since WESB and Vision were already existing and using XML, a solution using JSON would not be preferable.

Many of the advantages of JSON also prevail for XML. For example:

- Both have self describing data
- They provide structured data so that it is richer in information
- Internationalization (both use Unicode).

It is just a question of choosing the right tool for the right job. The right tool in this solution is SOAP with XML. SOAP is also a W3C recommendation. /12/

9 Result and discussion

The final result of this thesis work was a working system that can transfer information about solution items from Teamcenter to the TERPS endpoint database. The application and web service on the TERPS side also effectively notify Teamcenter of successful transfers and status reports. The solution is already partly implemented and it is verified that possible test data can be sent to TERPS endpoint database.

One thing worth mentioning is that I was working towards a development server and a copy of the TERPS database was used as a simulation of the real endpoint database. This was a very good thing, as it allowed me to more freely test different things without having to worry about destroying anything. It also gave me the freedom of inserting anything I wanted in the endpoint database (a copy of the real TEPRS database) for testing purposes.

During this project I have learned many new things about coding XML web services using the contract first technique, even though I had a lot of knowledge before. The thing most unfamiliar to me was changing the .wsdl files in order to make the SolutionItemService co-operable with the WESB side. Whether the solution of using an ASMX web service instead of a WCF service is the best way to solve this problem can be discussed, but based on the tools provided to accomplish this and the demands from WEBS, I think it is.

One thing that was very hard to keep in mind was trying to see the solution from a business point of view or data point of view. Something that seems as a good solution from a software point of view doesn't necessarily mean that it is a good solution if you look at it from another perspective.

Another hard thing during my assignment was to be able to guarantee that the transferred data from Teamcenter was exactly the same as the inserted data in the TERPS database. Something that was also very hard to foresee during the planning phase was the different business scenarios. For example, if a user enters the wrong information about a specific item in Teamcenter, there is no way to undo this if it has already been transferred, because it would mean a mismatch of the databases if the information is corrected "on the fly". The only solution to this would be to give the Teamcenter user a "regret" option, which would somehow regret the whole transfer. Currently the only way to correct a scenario like this is to create a new revision of the component in question and transfer it again.

Since I was not the only person working on the PDM Beta project, I quickly learned that good communication is very important. Because I was working with people in other countries we had to be sure we understood each other correctly to minimize the risk of losing precious time.

10 References

1. Backlund L. (2008) *TERPS product database.* Bachelors thesis in Industrial Management. Novia University of Applied Sciences, Vaasa

2. Bustamante LeRoux M. (2007). *Learning WCF*. (a.u.) O'Reilly Media

3. *Data quality* (n.d.) http://en.wikipedia.org/wiki/Data_quality (fetched 05.03.2013 at 17:48)

4. *Differences between ASMX and WCF Services.* (n.d.) http://msdn.microsoft.com/en-us/library/ff648181.aspx (fetched 28.12.2012 at 13:13)

5. Howard R, Microsoft Corporation (2001). *Web Services with ASP.NET* http://msdn.microsoft.com/en-us/library/ms972326.aspx (fetched 22.02.2013 at 20:35)

6. *JSON: The Fat-Free Alternative To XML* (n.d.) http://www.json.org/xml.html (fetched 21.03.2013 at 14:52)

7. *Maximum Capacity Specifications for SQL Server* (n.d.) http://msdn.microsoft.com/en-us/library/ms143432.aspx (fetched 22.02.2013 at 21:05)

8. *Microsoft Access SQL Reference* (n.d.) http://msdn.microsoft.com/en-us/library/office/bb259125(v=office.12).aspx (fetched 22.02.2013 at 21:40)

9. *Staging (data)* (n.d.) http://en.wikipedia.org/wiki/Staging_(data) (fetched 18.02.2013 at 19:47)

10. *Teamcenter software* (n.d.)
http://www.plm.automation.siemens.com/en_us/products/teamcenter/index.shtml
(fetched 8.11.2012 at 10.15)

11. Wärtsilä internal documentation.

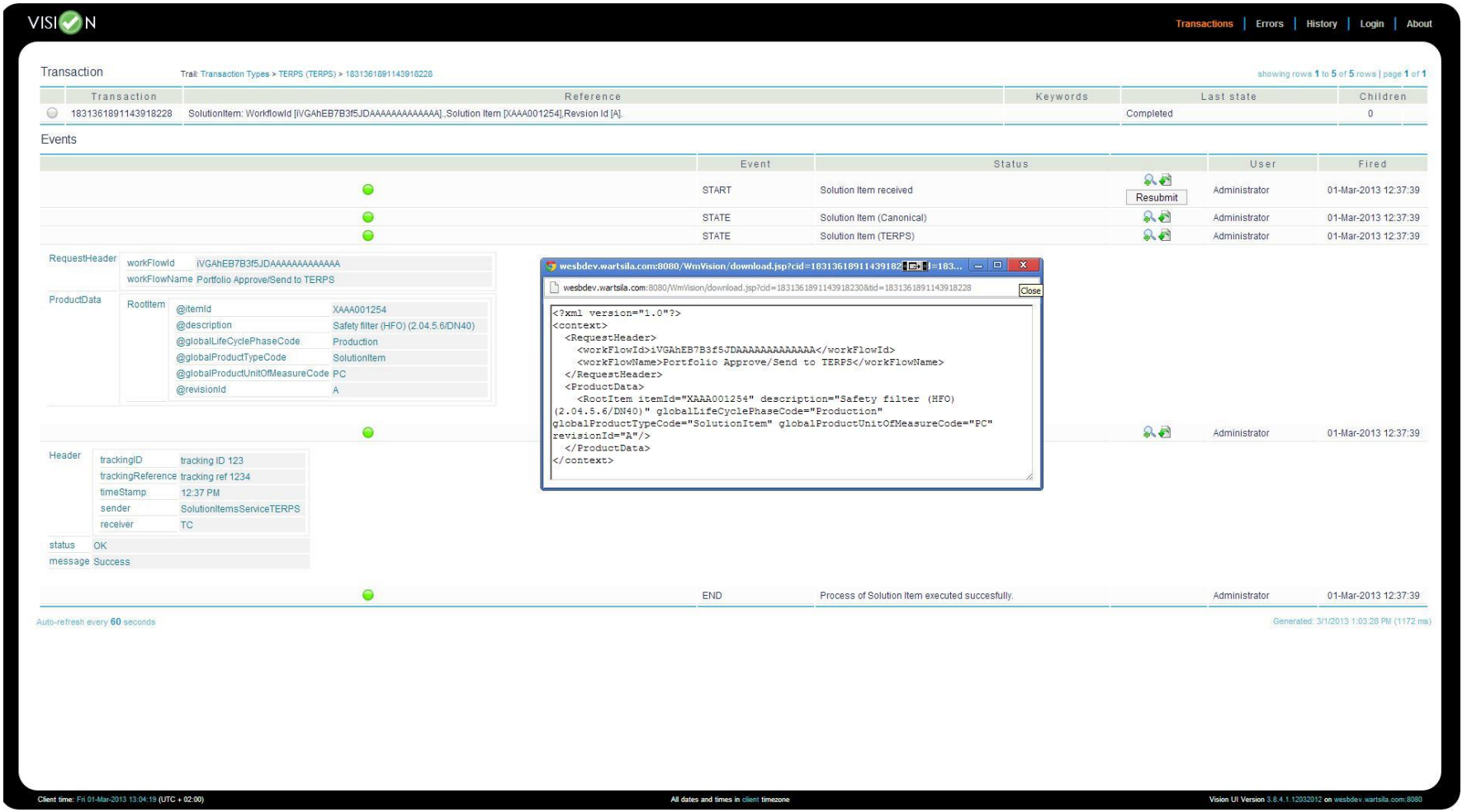
12. *W3C Latest SOAP versions* (n.d.)
http://www.w3.org/TR/soap/ (fetched 05.03 .2013 at 18.55)

Appendices

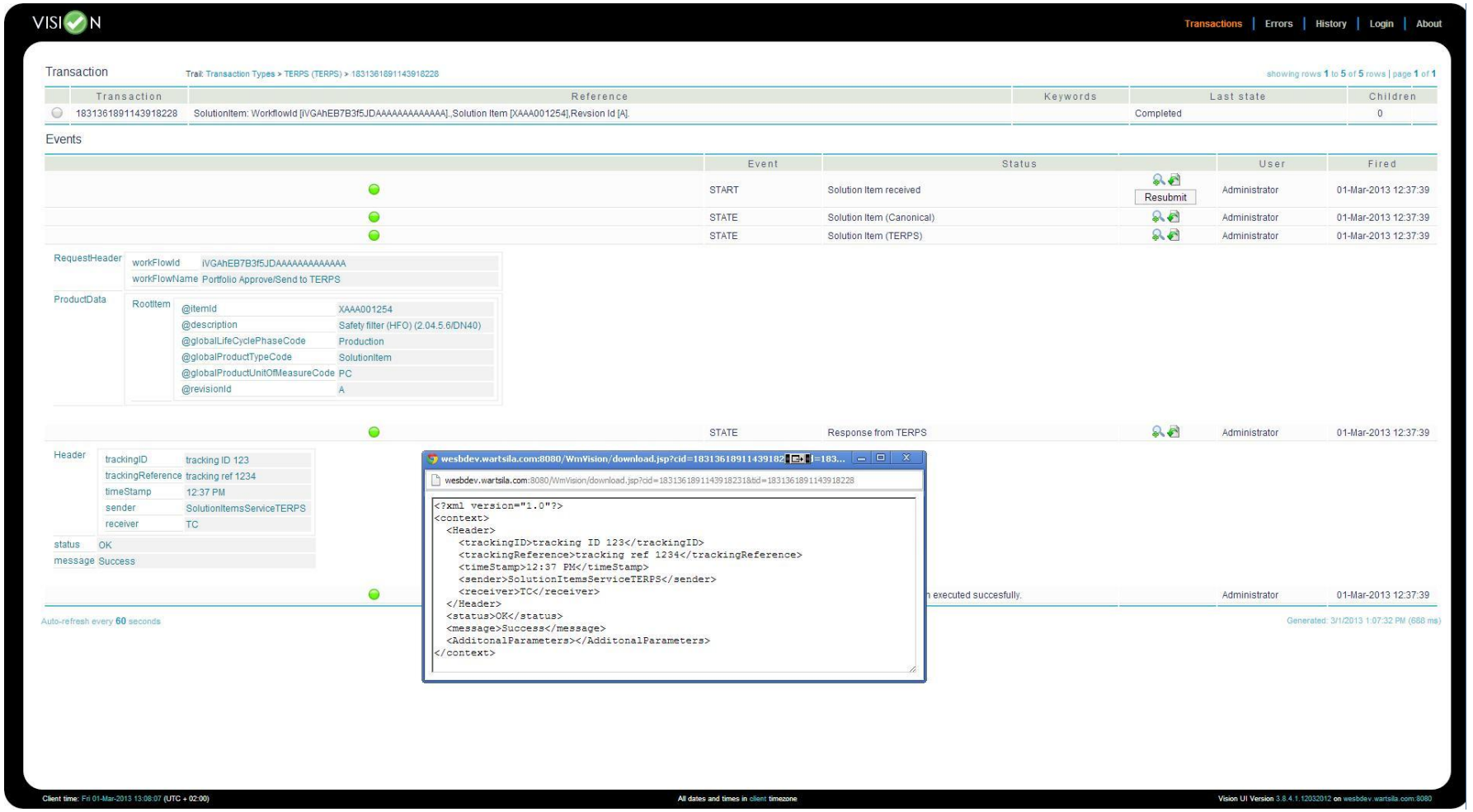
I. Solution item transfer displayed by Vision (with item data in XML)

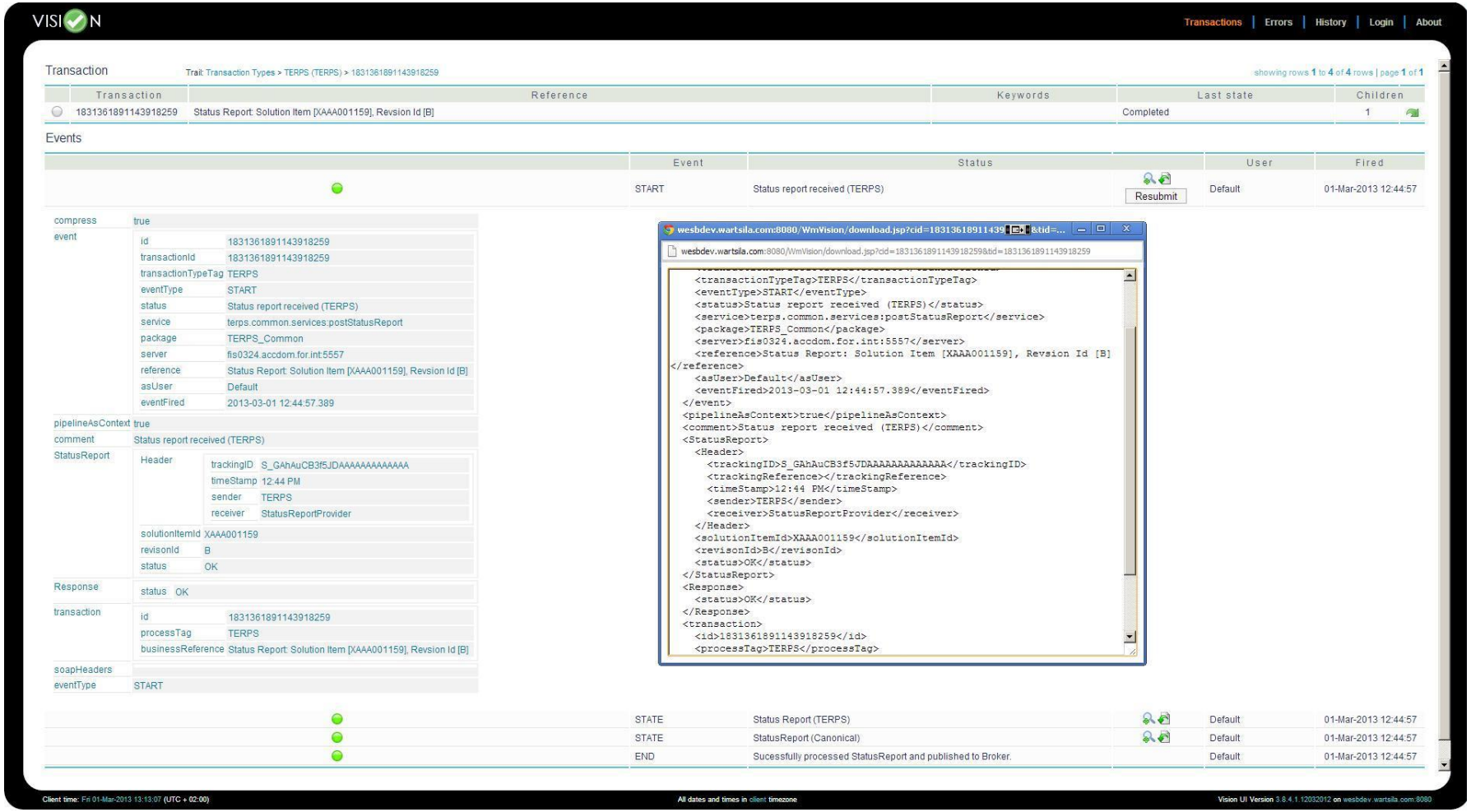
II. Solution item transfer displayed by Vision (with response data in XML)

III. Status report of transferred item displayed by Vision


VISION
Transactions | Errors | History | Login | About
Transaction
Trail: Transaction Types > TERPS (TERPS) > 1831361891143918228
showing rows 1 to 5 of 5 rows | page 1 of 1
Transaction
Reference
Keywords
Last state
Children
1831361891143918228
SolutionItem: WorkflowId [iVGAhEB7B3f5JDAAAAAAAAAAAA],Solution Item [XAAA001254],Revsion Id [A].
Completed
0
Events
Event
Status
User
Fired
START
Solution Item received
Resubmit
Administrator
01-Mar-2013 12:37:39
STATE
Solution Item (Canonical)
Administrator
01-Mar-2013 12:37:39
STATE
Solution Item (TERPS)
Administrator
01-Mar-2013 12:37:39
RequestHeader
workFlowId iVGAhEB7B3f5JDAAAAAAAAAAAA
workFlowName Portfolio Approve/Send to TERPS
ProductData
RootItem
@itemId XAAA001254
@description Safety filter (HFO) (2.04.5.6/DN40)
@globalLifeCyclePhaseCode Production
@globalProductTypeCode SolutionItem
@globalProductUnitOfMeasureCode PC
@revisionId A
wesbdev.wartsila.com:8080/WmVision/download.jsp?cid=1831361891143918230&tid=1831361891143918228
Close
<?xml version="1.0"?>
<context>
<RequestHeader>
<workFlowId>iVGAhEB7B3f5JDAAAAAAAAAAAA</workFlowId>
<workFlowName>Portfolio Approve/Send to TERPS</workFlowName>
</RequestHeader>
<ProductData>
<RootItem itemId="XAAA001254" description="Safety filter (HFO)
(2.04.5.6/DN40)" globalLifeCyclePhaseCode="Production"
globalProductTypeCode="SolutionItem" globalProductUnitOfMeasureCode="PC"
revisionId="A"/>
</ProductData>
</context>
Administrator
01-Mar-2013 12:37:39
Header
trackingID tracking ID 123
trackingReference tracking ref 1234
timeStamp 12:37 PM
sender SolutionItemsServiceTERPS
receiver TC
status OK
message Success
END
Process of Solution Item executed succesfully.
Administrator
01-Mar-2013 12:37:39
Auto-refresh every 60 seconds
Generated: 3/1/2013 1:03:28 PM (1172 ms)
Client time: Fri 01-Mar-2013 13:04:19 (UTC + 02:00)
All dates and times in client timezone
Vision UI Version 3.8.4.1.12032012 on wesbdev.wartsila.com:8080

Appendix I


VISION
Transactions | Errors | History | Login | About
Transaction
Trail: Transaction Types > TERPS (TERPS) > 1831361891143918228
showing rows 1 to 5 of 5 rows | page 1 of 1
Transaction
Reference
Keywords
Last state
Children
1831361891143918228
SolutionItem: WorkflowId [iVGAhEB7B3f5JDAAAAAAAAAAAA],Solution Item [XAAA001254],Revsion Id [A].
Completed
0
Events
Event
Status
User
Fired
START
Solution Item received
Resubmit
Administrator
01-Mar-2013 12:37:39
STATE
Solution Item (Canonical)
Administrator
01-Mar-2013 12:37:39
STATE
Solution Item (TERPS)
Administrator
01-Mar-2013 12:37:39
RequestHeader
workFlowId iVGAhEB7B3f5JDAAAAAAAAAAAA
workFlowName Portfolio Approve/Send to TERPS
ProductData
RootItem
@itemId XAAA001254
@description Safety filter (HFO) (2.04.5.6/DN40)
@globalLifeCyclePhaseCode Production
@globalProductTypeCode SolutionItem
@globalProductUnitOfMeasureCode PC
@revisionId A
STATE
Response from TERPS
Administrator
01-Mar-2013 12:37:39
Header
trackingID tracking ID 123
trackingReference tracking ref 1234
timeStamp 12:37 PM
sender SolutionItemsServiceTERPS
receiver TC
status OK
message Success
wesbdev.wartsila.com:8080/WmVision/download.jsp?cid=1831361891143918231&tid=1831361891143918228
<?xml version="1.0"?>
<context>
<Header>
<trackingID>tracking ID 123</trackingID>
<trackingReference>tracking ref 1234</trackingReference>
<timeStamp>12:37 PM</timeStamp>
<sender>SolutionItemsServiceTERPS</sender>
<receiver>TC</receiver>
</Header>
<status>OK</status>
<message>Success</message>
<AdditonalParameters></AdditonalParameters>
</context>
executed succesfully.
Administrator
01-Mar-2013 12:37:39
Auto-refresh every 60 seconds
Generated: 3/1/2013 1:07:32 PM (668 ms)
Client time: Fri 01-Mar-2013 13:08:07 (UTC + 02:00)
All dates and times in client timezone
Vision UI Version 3.8.4.1.12032012 on wesbdev.wartsila.com:8080


VISION
Transactions | Errors | History | Login | About
Transaction
Trail: Transaction Types > TERPS (TERPS) > 1831361891143918259
showing rows 1 to 4 of 4 rows | page 1 of 1
Transaction | Reference | Keywords | Last state | Children
1831361891143918259 | Status Report: Solution Item [XAAA001159], Revsion Id [B] | Completed | 1
Events
Event | Status | User | Fired
START | Status report received (TERPS) | Resubmit | Default | 01-Mar-2013 12:44:57
compress true
event
id 1831361891143918259
transactionId 1831361891143918259
transactionTypeTag TERPS
eventType START
status Status report received (TERPS)
service terps.common.services:postStatusReport
package TERPS_Common
server fis0324.accdom.for.int:5557
reference Status Report: Solution Item [XAAA001159], Revsion Id [B]
asUser Default
eventFired 2013-03-01 12:44:57.389
pipelineAsContext true
comment Status report received (TERPS)
StatusReport
Header
trackingID S_GAhAuCB3f5JDAAAAAAAAAAAA
timeStamp 12:44 PM
sender TERPS
receiver StatusReportProvider
solutionItemId XAAA001159
revisonId B
status OK
Response
status OK
transaction
id 1831361891143918259
processTag TERPS
businessReference Status Report: Solution Item [XAAA001159], Revsion Id [B]
soapHeaders
eventType START
wesbdev.wartsila.com:8080/WmVision/download.jsp?cid=1831361891143...&tid=...
wesbdev.wartsila.com:8080/WmVision/download.jsp?cid=1831361891143918259&tid=1831361891143918259
<transactionTypeTag>TERPS</transactionTypeTag>
<eventType>START</eventType>
<status>Status report received (TERPS)</status>
<service>terps.common.services:postStatusReport</service>
<package>TERPS_Common</package>
<server>fis0324.accdom.for.int:5557</server>
<reference>Status Report: Solution Item [XAAA001159], Revsion Id [B]
</reference>
<asUser>Default</asUser>
<eventFired>2013-03-01 12:44:57.389</eventFired>
</event>
<pipelineAsContext>true</pipelineAsContext>
<comment>Status report received (TERPS)</comment>
<StatusReport>
<Header>
<trackingID>S_GAhAuCB3f5JDAAAAAAAAAAAA</trackingID>
<trackingReference></trackingReference>
<timeStamp>12:44 PM</timeStamp>
<sender>TERPS</sender>
<receiver>StatusReportProvider</receiver>
</Header>
<solutionItemId>XAAA001159</solutionItemId>
<revisonId>B</revisonId>
<status>OK</status>
</StatusReport>
<Response>
<status>OK</status>
</Response>
<transaction>
<id>1831361891143918259</id>
<processTag>TERPS</processTag>
STATE | Status Report (TERPS) | Default | 01-Mar-2013 12:44:57
STATE | StatusReport (Canonical) | Default | 01-Mar-2013 12:44:57
END | Sucessfully processed StatusReport and published to Broker. | Default | 01-Mar-2013 12:44:57
Client time: Fri 01-Mar-2013 13:13:07 (UTC + 02:00)
All dates and times in client timezone
Vision UI Version 3.8.4.1.12032012 on wesbdev.wartsila.com:8080